In the Claims:

Please amend the Claims as follows:

1. (Twice Amended) An integrated circuit card for use with a terminal, comprising:

a communicator configured to communicate with the terminal;

a memory storing:

an application derived from a program written in a high level programming language format wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form, the converting step including at least one step selected from a group consisting of

C1

recording all jumps and their destinations in the original byte codes;

converting specific byte codes into equivalent generic byte codes or vice-versa;

modifying byte code operands from references using identifying strings to references using unique identifiers; and

renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation; and

an interpreter operable to interpret such an application derived from a program written in a high level programming language format; and

a processor coupled to the memory, the processor configured to use the interpreter to interpret the application for execution and to use the communicator to communicate with the terminal.

58

C2

29. (Amended) The integrated circuit card of claim 1, wherein the terminal has a wireless communication device and the [communictor]communicator a wireless transceiver for communicating with the wireless communication device.

31. (Twice Amended) A method for use with an integrated circuit card and a terminal, comprising:

storing an interpreter operable to interpret programs derived from programs written in a high level programming language and an application derived from a program written in a high level programming language format in a memory of the integrated circuit card wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form, the converting step including at least one step selected from a group consisting of

recording all jumps and their destinations in the original byte codes;

converting specific byte codes into equivalent generic byte codes or vice-versa;

modifying byte code operands from references using identifying strings to references using unique identifiers; and

renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation; and

using a processor of the integrated circuit card to use the interpreter to interpret the application for execution; and

using a communicator of the card when communicating between the processor and the terminal.

C3

91. (Twice Amended) An integrated circuit for use with a terminal, comprising:

C4

59

a communicator configured to communicate with the terminal;

a memory storing a first application that has been processed from a second application having a plurality of language elements including at least one string of characters, the string of characters being replaced in the first application by an identifier; and

C4

a processor coupled to the memory, the processor configured to use [the] an interpreter to interpret the first application for execution and to use the communicator to communicate with the terminal.

93. (Twice Amended) A method for use with an integrated circuit card and a terminal comprising:

processing a second application to create a first application, the second application having at least one programming element being a string of characters;

C5

replacing the string of characters of the first application with an identifier in the second application;

storing an interpreter and the first application in a memory of the integrated circuit card; and

using a processor of the integrated circuit card to use [an]the interpreter to interpret the first application for execution.

58 ~~95.~~ (Twice Amended) A microcontroller comprising:

a memory storing:

C6

a derivative application derived from an application having a class file format wherein the application is derived from an application having a class file format by first compiling the application having a class file format into a compiled form and then converting the compiled form into a converted form, the converting step including at least one step selected from a group consisting of

60

C

recording all jumps and their destinations in the original byte codes;

converting specific byte codes into equivalent generic byte codes or vice-versa;

C6 modifying byte code operands from references using identifying strings to references using unique identifiers; and

renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation, and

an interpreter configured to interpret applications derived from applications having a class file format; and

a processor coupled to the memory, the processor configured to use the interpreter to interpret the derivative application for execution.

C7 61 ~~98~~. (Amended) The microcontroller of claim 59 ~~96~~, wherein the terminal has a wireless [communictor]communicator and a wireless transceiver for communicating with the wireless communication device.

Please cancel Claims 101 through 104.

64 ~~105~~. (Twice Amended) An integrated circuit card for use with a terminal, comprising:

C8 a communicator configured to communicate with the terminal;

a memory storing:

applications, each application derived from applications having a high level programming language format, and

an interpreter operable to interpret applications derived from applications having a high level programming language format wherein the application is derived from a program written in a high

61

C

level programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form, the converting step including at least one step selected from a group consisting of

recording all jumps and their destinations in the original byte codes;

converting specific byte codes into equivalent generic byte codes or vice-versa;

modifying byte code operands from references using identifying strings to references using unique identifiers; and

renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation; and

C 8

a processor coupled to the memory, the processor configured to:

a.) use the interpreter to interpret the applications for execution,

b.) use the interpreter to create a firewall to isolate the applications from each other, and

c.) use the communicator to communicate with the terminal.

Sub D4

106. (Amended) A microcontroller having a set of resource constraints and comprising:

a memory, and

an interpreter loaded in memory and operable within the set of resource constraints,

the microcontroller having: at least one application loaded in the memory to be interpreted by the interpreter, wherein the at least one application is generated by a programming environment comprising:

a) a compiler for compiling application source programs written in high level language source code form into a compiled form, and

62

C8

b) a converter for post processing the compiled form into a minimized form suitable for interpretation within the set of resource constraints by the interpreter.

Sub D5

120. (Amended) A method of programming a microcontroller having a memory and a processor operating according to a set of resource constraints, the method comprising the steps of:

C9

inputting an application program in a first programming language;

compiling the application program in the first programming language into a first intermediate code associated with the first programming language, wherein the first intermediate code being interpretable by at least one first intermediate code virtual machine;

converting the first intermediate code into a second intermediate code; wherein the second intermediate code is interpretable within the set of resource constraints by at least one second intermediate code virtual machine; and

loading the second intermediate code into the memory of the microcontroller.

127. (Not amended but reproduced here for the Examiner's convenience.) A microcontroller operable to execute derivative programs which are derivatives of programs written in an interpretable programming language having a memory and an interpreter, the microcontroller comprising:

(a) the microcontroller operating within a set of resource constraints including the memory being of insufficient size to permit interpretation of programs written in the interpretable programming language; and

(b) the memory containing an interpreter operable to interpret the derivative programs written in the derivative of the interpretable language wherein a derivative of a program written in the interpretable programming language is derived from the program written in the interpretable programming language by applying at least one rule selected from a set of rules including:

(1) mapping strings to identifiers;

(2) performing security checks prior to or during interpretation;

(3) performing structural checks prior to or during interpretation; and

(4) performing semantic checks prior to or during interpretation.

Please add the following new claims:

Sub D7

-- 137. An integrated circuit card for use with a terminal, comprising:

a communicator configured to communicate with the terminal;

a memory storing:

an application derived from a program written in a high level programming language format wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form, the converting step including modifying byte code operands from references using identifying strings to references using unique identifiers; and

C10

an interpreter operable to interpret such an application derived from a program written in a high level programming language format; and

a processor coupled to the memory, the processor configured to use the interpreter to interpret the application for execution and to use the communicator to communicate with the terminal.

138. The integrated circuit card of Claim 137 wherein the converting step further comprises:

recording all jumps and their destinations in the original byte codes;

converting specific byte codes into equivalent generic byte codes or vice-versa; and renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation.

C10

139. A method for use with an integrated circuit card and a terminal, comprising:

storing an interpreter operable to interpret programs derived from programs written in a high level programming language and an application derived from a program written in a high level programming language format in a memory of the integrated circuit card wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form, the converting step including modifying byte code operands from references using identifying strings to references using unique identifiers; and

using a processor of the integrated circuit card to use the interpreter to interpret the application for execution; and

using a communicator of the card when communicating between the processor and the terminal.

Sub E1

140. The method of Claim 139 wherein the converting step further comprises:

recording all jumps and their destinations in the original byte codes;

converting specific byte codes into equivalent generic byte codes or vice-versa; and

renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation.

Sub D8

141. An integrated circuit card for use with a terminal, comprising:

a communicator configured to communicate with the terminal;

a memory storing:

C

applications, each application derived from applications having a high level programming language format, and

an interpreter operable to interpret applications derived from applications having a high level programming language format wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form, the converting step including modifying byte code operands from references using identifying strings to references using unique identifiers; and

e10

a processor coupled to the memory, the processor configured to:

a.) use the interpreter to interpret the applications for execution,

b.) use the interpreter to create a firewall to isolate the applications from each other, and

c.) use the communicator to communicate with the terminal.

142. The integrated circuit card of Claim 141 wherein the interpreter is further operable to interpret applications derived using a converting step including:

recording all jumps and their destinations in the original byte codes;

converting specific byte codes into equivalent generic byte codes or vice-versa; and

renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation.

143. A microcontroller operable to execute derivative programs which are derivatives of programs written in an interpretable programming language having a memory and an interpreter, the microcontroller comprising:

the microcontroller operating within a set of resource constraints including the memory being of insufficient size to permit interpretation of programs written in the interpretable programming language; and

the memory containing an interpreter operable to interpret the derivative programs written in the derivative of the interpretable language wherein a derivative of a program written in the interpretable programming language is derived from the program written in the interpretable programming language by mapping strings to identifiers.

C10

144. A microcontroller comprising:

a memory storing:

a derivative application derived from an application having a class file format wherein the application is derived from an application having a class file format by first compiling the application having a class file format into a compiled form and then converting the compiled form into a converted form, the converting step including:

recording all jumps and their destinations in the original byte codes;

converting specific byte codes into equivalent generic byte codes or vice-versa; and

renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation, and

an interpreter configured to interpret applications derived from applications having a class file format; and

a processor coupled to the memory, the processor configured to use the interpreter to interpret the derivative application for execution.--